

**LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE**



Oleh:

RIRIN CITRA LESTARI

NIM. 2310817120012

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
2025**

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

Laporan Akhir Praktikum Pemrograman Mobile ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Akhir Praktikum ini dikerjakan oleh:

Nama Praktikan : Ririn Citra Lestari
NIM : 2310817120012

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Muhammad Raka Azwar
NIM. 2210817210012

EKA SETYA WIJAYA, S.T., M.Kom.
NIP. 198205082008011010

DAFTAR ISI

| | |
|---|----|
| LEMBAR PENGESAHAN | 2 |
| DAFTAR ISI..... | 3 |
| DAFTAR GAMBAR..... | 5 |
| DAFTAR TABEL..... | 6 |
| MODUL 1 : ANDROID BASIC WITH COMPOSE | 8 |
| SOAL 1 | 8 |
| A. Source Code..... | 10 |
| B. Output Program | 16 |
| C. Pembahasan | 19 |
| MODUL 2 : ANDROID LAYOUT WITH COMPOSE | 20 |
| SOAL 1..... | 20 |
| A. Source Code..... | 21 |
| B. Output Program | 33 |
| C. Pembahasan | 35 |
| SOAL 2 | 36 |
| MODUL 3 : BUILD A SCROLLABLE LIST | 37 |
| SOAL 1 | 37 |
| A. Source Code..... | 39 |
| B. Output Program | 53 |
| C. Pembahasan | 56 |
| SOAL 2 | 58 |
| MODUL 4 : VIEW MODEL DAN DEBUGGING..... | 59 |
| SOAL 1 | 59 |
| A. Source Code..... | 59 |
| B. Output Program | 76 |
| C. Pembahasan | 79 |
| SOAL 2 | 80 |
| MODUL 5 : FUNCTION DAN DATABASE..... | 81 |
| SOAL 1 | 81 |

| | | |
|------------------|----------------------|----|
| A. | Source Code..... | 82 |
| B. | Output Program | 92 |
| C. | Pembahasan | 2 |
| Tautan Git | | 3 |

DAFTAR GAMBAR

| | |
|---|----|
| Gambar 1. Soal 1 Modul 1 | 8 |
| Gambar 2. Soal 1 Modul 1 | 9 |
| Gambar 3. Screenshot Hasil Jawaban Soal 1 Modul 1 | 16 |
| Gambar 4. Screenshot hasil jawaban Soal 1 Modul 1 | 16 |
| Gambar 5. Screenshot Hasil Jawaban Soal 1 Modul 1 | 17 |
| Gambar 6. Screenshot Hasil Jawaban Soal 1 Modul 1 | 17 |
| Gambar 7. Screenshot Hasil Jawaban Soal 1 Modul 1 | 18 |
| Gambar 8. Screenshot Hasil Jawaban Soal 1 Modul 1 | 18 |
| Gambar 9. Soal 1 Modul 2 | 20 |
| Gambar 10. Soal 1 Modul 2 | 21 |
| Gambar 11. Soal 1 Modul 2 | 21 |
| Gambar 12. Screenshot Soal 1 Modul 2 | 33 |
| Gambar 13. Screenshot Soal 1 Modul 2 | 33 |
| Gambar 14. Screenshot Soal 1 Modul 2 | 34 |
| Gambar 15. Screenshot Soal 1 Modul 2 | 34 |
| 8. Aplikasi berbasis XML harus menggunakan ViewBinding | |
| Gambar 16. Soal 1 Modul 3 | 37 |
| Gambar 17. Soal 1 Modul 3 | 37 |
| Gambar 18. Soal 1 Modul 3 | 38 |
| Gambar 19. Screenshot Jawaban Hasil Soal 1 Modul 3 | 53 |
| Gambar 20. Screenshot Jawaban Hasil Soal 1 Modul 3 | 54 |
| Gambar 21. Screenshot Jawaban Hasil Soal 1 Modul 3 | 56 |
| Gambar 22. Screenshot Hasil Jawaban Soal 1 Modul 3 | 56 |
| Gambar 23. Soal 1 Modul 4 | 59 |
| Gambar 24. Breakpoint (Compose) | 76 |
| Gambar 25. Step Over (F8) Digunakan (Compose) | 76 |
| Gambar 26. Step Out (Shift + F8) Keluar Dari Fungsi (Compose) | 76 |
| Gambar 27. Resume (F9) Untuk Melanjutkan Eksekusi (Compose) | 77 |
| Gambar 28. Breakpoint (XML) | 77 |
| Gambar 29. Debug (XML) | 77 |
| Gambar 30. Step Into (F7) Masuk Ke Fungsi (XML) | 78 |
| Gambar 31. Step Over (F8) Digunakan (XML) | 78 |
| Gambar 32. Step Out (Shift + F8) Keluar Dari Fungsi (XML) | 78 |
| Gambar 33. Screenshot Hasil Jawaban Soal 1 Modul 5 | 92 |
| Gambar 34. Screenshot Hasil Jawaban Soal 1 Modul 5 | 93 |
| Gambar 35. Screenshot Hasil Jawaban Soal 1 Modul | 94 |

DAFTAR TABEL

| | |
|---|----|
| Tabel 1.Source Code Soal 1 Modul 1 | 13 |
| Tabel 2.Source Code Soal 1 Modul 1 | 15 |
| Tabel 3. Source Code Soal 1 Modul 2 | 27 |
| Tabel 4.Source Code Soal 1 Modul 2 | 28 |
| Tabel 5.Source Code Soal 1 Modul 2 | 30 |
| Tabel 6.Source Code Soal 1 Modul 2 | 32 |
| Tabel 7. Source Code Soal 1 Modul 3 | 40 |
| Tabel 8.Source Code Soal 1 Modul 3 | 42 |
| Tabel 9.Source Code Soal 1 Modul 3 | 42 |
| Tabel 10.Source Code Soal 1 Modul 3 | 43 |
| Tabel 11.Source Code Soal 1 Modul 3 | 45 |
| Tabel 12.Source Code Soal 1 Modul 3 | 46 |
| Tabel 13.Source Code Soal 1 Modul 3 | 47 |
| Tabel 14.Source Code Soal 1 Modul 3 | 48 |
| Tabel 15.Source Code Soal 1 Modul 3 | 49 |
| Tabel 16.Source Code Soal 1 Modul 3 | 50 |
| Tabel 17.Source Code Soal 1 Modul 3 | 50 |
| Tabel 18.Source Code Soal 1 Modul 3 | 51 |
| Tabel 19.Source Code Soal 1 Modul 3 | 52 |
| Tabel 20.Source Code Soal 1 Modul 4 | 60 |
| Tabel 21.Source Code Soal 1 Modul 4 | 62 |
| Tabel 22.Source Code Soal 1 Modul 4 | 63 |
| Tabel 23.Source Code Soal 1 Modul 4 | 64 |
| Tabel 24.Source Code Soal 1 Modul 4 | 66 |
| Tabel 25.Source Code Soal 1 Modul 4 | 67 |
| Tabel 26.Source Code Soal 1 Modul 4 | 68 |
| Tabel 27.Source Code Soal 1 Modul 4 | 68 |
| Tabel 28.Source Code Soal 1 Modul 4 | 68 |
| Tabel 29.Source Code Soal 1 Modul 4 | 69 |
| Tabel 30.Source Code Soal 1 Modul 4 | 71 |
| Tabel 31.Source Code Soal 1 Modul 4 | 71 |
| Tabel 32.Source Code Soal 1 Modul 4 | 72 |
| Tabel 33.Source Code Soal 1 Modul 4 | 73 |
| Tabel 34.Source Code Soal 1 Modul 4 | 74 |
| Tabel 35.Source Code Soal 1 Modul 4 | 74 |
| Tabel 36.Source Code Soal 1 Modul 4 | 75 |
| Tabel 37.Source Code Soal 1 Modul 5 | 82 |
| Tabel 38.Source Code Soal 1 Modul 5 | 83 |
| Tabel 39.Source Code Soal 1 Modul 5 | 83 |

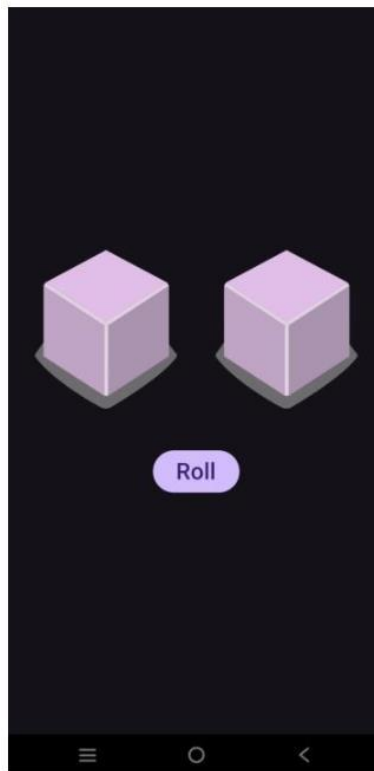
| | |
|---|----|
| Tabel 40.Source Code Soal 1 Modul 5 | 84 |
| Tabel 41.Source Code Soal 1 Modul 5 | 85 |
| Tabel 42.Source Code Soal 1 Modul 5 | 87 |
| Tabel 43.Source Code Soal 1 Modul 5 | 87 |
| Tabel 44.Source Code Soal 1 Modul 5 | 88 |
| Tabel 45.Source Code Soal 1 Modul 5 | 88 |
| Tabel 46.Source Code Soal 1 Modul 5 | 88 |
| Tabel 47.Source Code Soal 1 Modul 5 | 89 |
| Tabel 48.Source Code Soal 1 Modul 5 | 90 |
| Tabel 49.Source Code Soal 1 Modul 5 | 90 |
| Tabel 50.Source Code Soal 1 Modul 5 | 90 |
| Tabel 51.Source Code Soal 1 Modul 5 | 91 |
| Tabel 52.Source Code Soal 1 Modul 5 | 91 |

MODUL 1 : ANDROID BASIC WITH COMPOSE

SOAL 1

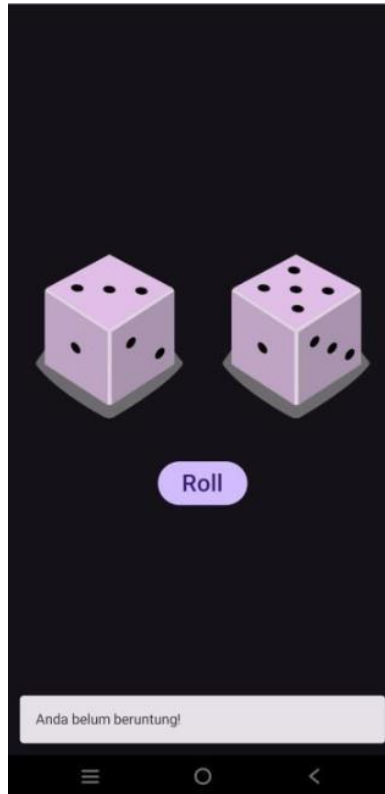
Buatlah sebuah aplikasi yang dapat menampilkan 2 buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

1. Tampilan awal aplikasi setelah dijalankan akan menampilkan 2 buah dadu kosong seperti dapat dilihat pada Gambar 1



Gambar 1.Soal 1 Modul 1

2. Setelah user menekan tombol “Roll” maka masing-masing dadu akan memperlihatkan sisi dadunya dengan angka antara 1 s/d 6. Apabila user mendapatkan nilai dadu yang berbeda antara Dadu 1 dengan Dadu 2, maka aplikasi akan menampilkan pesan “Anda belum beruntung!” seperti yang dapat dilihat pada Gambar 2.



Gambar 2. Soal 1 Modul 1

3. Apabila user mendapatkan nilai dadu yang sama antara Dadu 1 dan Dadu 2 atau nilai double, maka aplikasi akan menampilkan pesan “Selamat, anda dapat dadu double!” seperti yang dapat dilihat pada Gambar 3.
4. Buatlah aplikasi tersebut menggunakan XML dan Jetpack Compose.

5. Upload aplikasi yang telah anda buat ke dalam repository GitHub ke dalam folder Modul 1 dalam bentuk Project. Jangan lupa untuk melakukan Clean Project sebelum mengupload pekerjaan anda pada repository.

Untuk gambar dadu dapat didownload pada link berikut:

https://drive.google.com/file/d/14V3qXGdFnuoYN4AGd_9SgFh8kw8X9ySm/view

A. Source Code

Compose

MainActivity.kt

```
package com.example.doubledice

import android.os.Bundle
import androidx.activity.ComponentActivity
import
androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import
androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import
androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import
androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import
androidx.compose.foundation.layout.width
import androidx.compose.material3.Button
import
```

```

androidx.compose.material3.Text
import
androidx.compose.runtime.Composable
import
androidx.compose.runtime.getValue
import
androidx.compose.runtime.mutableIntStateOf
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import
androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.painterResource
import
androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            DiceApp()
        }
    }
}

@Composable
fun DiceApp() {
    var          dice1          by
                remember

```

```

{ mutableIntStateOf(R.drawable.dice_0) }

    var          dice2          by
        remember

{ mutableIntStateOf(R.drawable.dice_0) }

    var resultText by remember { mutableStateOf("") }

    val diceImages = listOf(
        R.drawable.dice_1,
        R.drawable.dice_2,
        R.drawable.dice_3,
        R.drawable.dice_4,
        R.drawable.dice_5,
        R.drawable.dice_6
    )

    Column(
        horizontalAlignment =
        Alignment.CenterHorizontally, verticalArrangement
        = Arrangement.Center,
        modifier = Modifier.fillMaxSize().padding(16.dp)
    ) {
        Row {
            Image(painter = painterResource(id =
                dice1), contentDescription =
null, modifier = Modifier.size(100.dp))
            Spacer(modifier = Modifier.width(16.dp))
            Image(painter = painterResource(id =
                dice2),
contentDescription = null, modifier =
Modifier.size(100.dp))
        }
    }

```

```

        Spacer(modifier = Modifier.height(24.dp))

        Button(onClick = {
            dice1 =
            diceImages.random() dice2
            = diceImages.random()
            resultText = if (dice1 == dice2) "Selamat, anda
dapat dadu double!" else "Anda belum beruntung!"
        }) {
            Text("Roll")
        }

        Spacer(modifier = Modifier.height(16.dp))
        Text(text = resultText, fontSize = 18.sp)
    }
}

@Preview(showBackground = true)
@Composable
fun DiceAppPreview() {
    DiceApp()
}

```

Tabel 1.Source Code Soal 1 Modul 1

XML

XmlActivity.kt

```

package com.example.doubledice.ui.theme

import android.os.Bundle
import
android.widget.Button
import android.widget.ImageView

```

```

import android.widget.TextView
import
androidx.appcompat.app.AppCompatActivity
import com.example.doubledice.R

class MainActivity : AppCompatActivity() {
    private lateinit var diceImage1:
    ImageView private lateinit var
    diceImage2: ImageView private lateinit
    var resultText: TextView

    private val diceImages = listOf(
        R.drawable.dice_1,
        R.drawable.dice_2,
        R.drawable.dice_3,
        R.drawable.dice_4,
        R.drawable.dice_5,
        R.drawable.dice_6
    )

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        diceImage1 =
        findViewById(R.id.diceImage1)
        diceImage2 =
        findViewById(R.id.diceImage2)
        resultText =
        findViewById(R.id.resultText)

        val rollButton: Button =
        findViewById(R.id.rollButton)
    }
}

```

```

        rollButton.setOnClickListener {
            rollDice()
        }
    }

    private fun rollDice() {
        val dice1 =
            diceImages.random() val
            dice2 = diceImages.random()

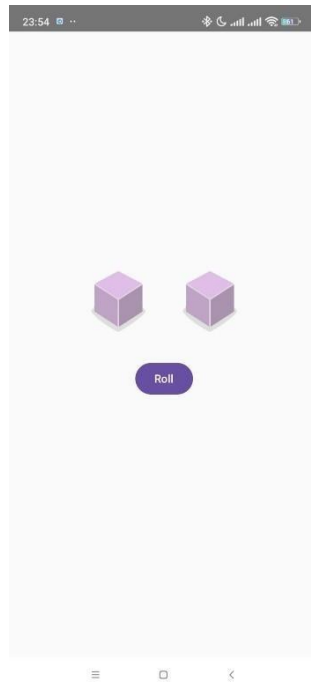
        diceImage1.setImageResource(dice1)
        diceImage2.setImageResource(dice2)

        resultText.text = if (dice1 ==
            dice2) { "Selamat, anda dapat
            dadu double!"
        } else {
            "Anda belum beruntung!"
        }
    }
}

```

Tabel 2.Source Code Soal 1 Modul 1

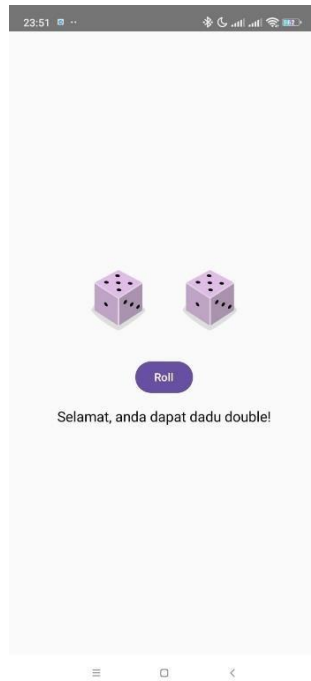
B. Output Program



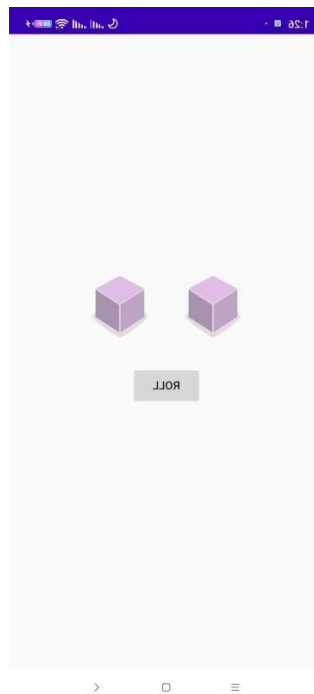
Gambar 3. Screenshot Hasil Jawaban Soal 1 Modul 1



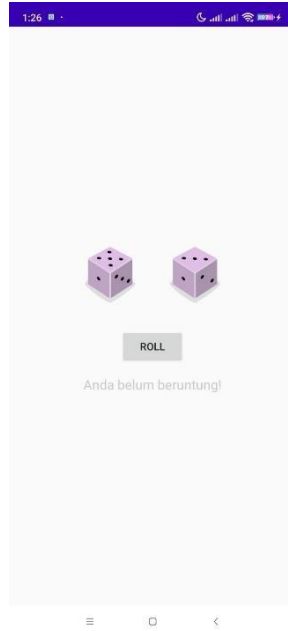
Gambar 4. Screenshot hasil jawaban Soal 1 Modul 1



Gambar 5.Screeshot Hasil Jawaban Soal 1 Modul 1



Gambar 6.Screeshot Hasil Jawaban Soal 1 Modul 1



Gambar 7.Screenshot Hasil Jawaban Soal 1 Modul 1



Gambar 8.Screenshot Hasil Jawaban Soal 1 Modul 1

C. Pembahasan

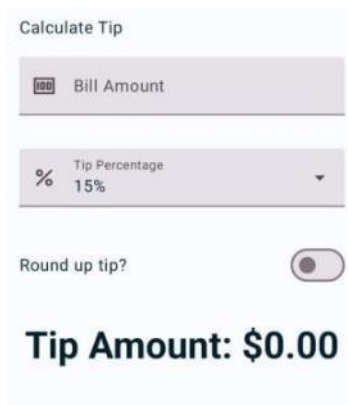
Kode di atas merupakan implementasi dari aplikasi lempar dua dadu (Double Dice) dalam dua pendekatan berbeda pada Android: menggunakan Jetpack Compose (pada MainActivity.kt) dan UI berbasis XML tradisional (pada XmlActivity.kt dan activity_main.xml). Pada pendekatan Jetpack Compose (MainActivity.kt), fungsi DiceApp() membentuk UI secara deklaratif menggunakan elemen-elemen seperti Column, Row, Image, dan Button. Dua variabel dice1 dan dice2 digunakan untuk menyimpan gambar dadu secara dinamis. Saat tombol “Roll” ditekan, dua gambar dadu diubah secara acak dari daftar diceImages, lalu dicek apakah nilainya sama. Jika iya, maka akan ditampilkan teks “Selamat, anda dapat dadu double!”; jika tidak, teks menunjukkan bahwa pemain belum beruntung. Sementara pada pendekatan tradisional XML (XmlActivity.kt dan activity_main.xml), tampilan UI dibuat di file XML (activity_main.xml) menggunakan LinearLayout yang menyusun dua ImageView untuk gambar dadu, sebuah Button untuk melakukan lemparan, dan sebuah TextView untuk menampilkan hasil. Di dalam kelas MainActivity pada XmlActivity.kt, elemenelemen UI dihubungkan menggunakan findViewById. Ketika tombol diklik, metode rollDice() dijalankan untuk memilih dua gambar dadu secara acak dari list, mengatur ulang tampilan gambar, dan menampilkan pesan sesuai hasil. Kedua pendekatan ini menghasilkan fungsionalitas yang sama, namun Compose memberikan cara yang lebih modern dan deklaratif dalam membangun UI di Android, sementara XML memberikan kontrol yang lebih eksplisit dan tradisional seperti yang digunakan sebelum Compose hadir

MODUL 2 : ANDROID LAYOUT WITH COMPOSE

SOAL 1

Buatlah sebuah aplikasi kalkulator tip menggunakan XML dan Jetpack Compose yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

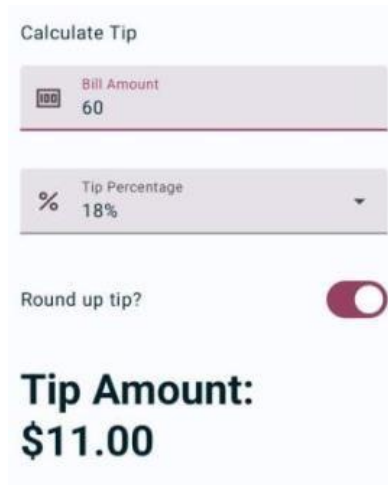
- Input biaya layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.
- Pilihan persentase tip: Pengguna dapat memilih persentase tip yang diinginkan.
- Pengaturan pembulatan tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.
- Tampilan hasil: Aplikasi akan menampilkan jumlah tip yang harus dibayar secara langsung setelah pengguna memberikan input.



Gambar 9.Soal 1 Modul 2



Gambar 10. Soal 1 Modul 2



Gambar 11. Soal 1 Modul 2

A. Source Code

Compose

MainActivity.kt

```
package com.example.tipcalculation

import android.os.Bundle

import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.* import
import androidx.compose.ui.Alignment      import
import androidx.compose.ui.Modifier

import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.unit.dp
```

```

import
androidx.compose.foundation.text.KeyboardOptions
import java.text.NumberFormat

import kotlin.math.ceil

import androidx.compose.ui.tooling.preview.Preview as Preview1
import androidx.compose.material.icons.Icons

class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContent {

            TipCalculatorApp()

        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable

fun TipCalculatorApp() {

    var amountInput by remember { mutableStateOf("") }

    var tipPercent by remember {
        mutableDoubleStateOf(0.18) } var roundUp by
        remember { mutableStateOf(true) }

```

```

val amount = amountInput.toDoubleOrNull() ?:
0.0 var tip = amount * tipPercent

if (roundUp) tip = ceil(tip)

val formattedTip = "$" +
String.format("%.2f", tip) val
percentageOptions = listOf(0.15, 0.18, 0.20)
val expanded = remember {
mutableStateOf(false) }

Column(modifier = Modifier

.fillMaxWidth()

.padding(24.dp)) {

    Text(text = "Calculate Tip", style =
MaterialTheme.typography.titleMedium)

    Spacer(Modifier.height(8.dp))

    OutlinedTextField(

        value = amountInput,

```

```

        onValueChange = { amountInput = it
        }, label = { Text("Bill Amount") },
keyboardOptions=
    KeyboardOptions(keyboardType =
        modifier = Modifier.fillMaxWidth()
    )
    Spacer(Modifier.height(16.dp))

    ExposedDropDownMenuBox(
        expanded =
        expanded.value,
        onExpandedChange = { expanded.value = !expanded.value }
    ) {
        OutlinedTextField(
            readOnly = true,
            value = "${(tipPercent *
            100).toInt()}%", onValueChange = {},
            label = { Text("Tip Percentage") },
            trailingIcon = {
                ExposedDropDownMenuDefaults.TrailingIcon(expanded
                = expanded.value)
            },
            modifier = Modifier

```



```

        .menuAnchor()
        .fillMaxWidth()
    )

    ExposedDropdownMenu(
        expanded = expanded.value,
        onDismissRequest = { expanded.value = false }
    ) {
        percentageOptions.forEach { option ->
            DropdownMenuItem(
                text = { Text("${(option * 100).toInt()}%") },
                onClick = {
                    tipPercent = option
                    expanded.value = false
                }
            )
        }
    }
}

Spacer(Modifier.height(16.dp))

Row(

```

```

        verticalAlignment =
        Alignment.CenterVertically,    modifier =
        Modifier.fillMaxWidth()

    ) {
        Text("Round up tip?")
        Spacer(modifier =
        Modifier.weight(1f)) Switch(

            checked = roundUp,
            onChangeChange = { roundUp =
            it }

        )
    }

    Spacer(Modifier.height(32.dp))

    Text(
        text = "Tip Amount:",
        style = MaterialTheme.typography.headlineSmall
    )
    Text(
        text = formattedTip,
        style = MaterialTheme.typography.headlineSmall,
        color = MaterialTheme.colorScheme.onBackground
    )

```

```

    }
}

@Preview1(showBackground = true)
@Composable
fun TipCalculatorAppPreview() {
    TipCalculatorApp()
}

```

Tabel 3. Source Code Soal 1 Modul 2

XML

XmlActivity.kt

```

package com.example.tipcalculation

import android.annotation.SuppressLint
import android.os.Bundle
import android.widget.*
import androidx.appcompat.app.AppCompatActivity
import java.text.NumberFormat
import kotlin.math.ceil

class XmlActivity : AppCompatActivity() {

    private lateinit var costEditText: EditText
    private lateinit var tipResult: TextView
    private lateinit var tipOptions: RadioGroup
    @SuppressLint("UseSwitchCompatOrMaterialCod
e") private lateinit var roundUpSwitch:
Switch private lateinit var
switchToCompose: Button

```

```

        override fun onCreate(savedInstanceState: Bundle?)
        {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_xml)

            // Inisialisasi view
            costEditText = findViewById(R.id.costOfServiceEditText)
            tipResult = findViewById(R.id.tipResult)
            tipOptions = findViewById(R.id.tipOptions)
            roundUpSwitch = findViewById(R.id.roundUpSwitch)

            // Hitung tip saat user menekan enter
            costEditText.setOnEditorActionListener { _, _, _
            ->
                calculateTip()

            true
        }
    }

    private fun calculateTip() {
        val cost = costEditText.text.toString().toDoubleOrNull() if
        (cost == null) {
            tipResult.text = "" return
        }

        val tipPercentage = when (tipOptions.checkedRadioButtonId) {
            R.id.optionTwenty -> 0.20
            R.id.optionEighteen -> 0.18
            R.id.optionFifteen -> 0.15
            else -> 0.15
        }

        var tip = cost * tipPercentage if
        (roundUpSwitch.isChecked) {
            tip = ceil(tip)
        }

        Val formattedTip
        NumberFormat.getCurrencyInstance().format(tip)
        tipResult.text = getString(R.string.tip_amount, formattedTip)
    }
}

```

Tabel 4.Source Code Soal 1 Modul 2

ActivityMain.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    "
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="24dp">

        <EditText
            android:id="@+id/costOfServiceEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:autofillHints=""
            android:hint="@string/enter_cost_of_service"
            android:inputType="numberDecimal"
            android:minHeight="48dp" android:padding="12dp"
            />

        <RadioGroup
            android:id="@+id/tipOptions"

            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:layout_marginTop="16dp">

            <RadioButton
                android:id="@+id/optionTwenty"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/option_twenty"
                android:minHeight="48dp"
                android:padding="12dp" />

            <RadioButton
                android:id="@+id/optionEighteen"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/option_eighteen"
```

```

        android:minHeight="48dp"
        android:padding="12dp" />

        <RadioButton
            android:id="@+id/optionFifteen"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/option_fifteen"
            android:minHeight="48dp"
            android:padding="12dp" />
    </RadioGroup>

    <Switch
        android:id="@+id/roundUpSwitch"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/round_up"
        android:layout_marginTop="44dp"
        android:padding="12dp"
        tools:ignore="UseSwitchCompatOrMaterialXml"
    />

    <TextView
        android:id="@+id/tipResult"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/tip_default"
        android:textSize="20sp"
        android:layout_marginTop="16dp" />

    <Button
        android:id="@+id/calculateButton"
        android:layout_width="147dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Calculate" />
</LinearLayout>
</ScrollView>

```

Tabel 5.Source Code Soal 1 Modul 2

MainActivity.kt

```

package com.example.tipcalculation

import android.os.Bundle

```

```

import
android.view.inputmethod.EditorInfo
import android.widget.*
import
androidx.appcompat.app.AppCompatActivity
import java.text.NumberFormat
import kotlin.math.ceil

class MainActivity : AppCompatActivity() {

    private lateinit var costEditText: EditText
    private lateinit var tipResult: TextView
    private lateinit var tipOptions: RadioGroup
    private lateinit var roundUpSwitch: Switch
    private lateinit var calculateButton:
    Button

    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_xml)

        costEditText = findViewById(R.id.costOfServiceEditText)
        tipResult = findViewById(R.id.tipResult)
        tipOptions = findViewById(R.id.tipOptions)
        roundUpSwitch =
        findViewById(R.id.roundUpSwitch)
        calculateButton =
        findViewById(R.id.calculateButton)

        //      Listener      tombol      Calculate
        calculateButton.setOnClickListener {
            calculateTip()
        }

        //      Listener      keyboard      Done
        costEditText.setOnEditorActionListener { _, actionId, _
        ->
            if (actionId == EditorInfo.IME_ACTION_DONE) {
                calculateTip()
                true
            } else {
                false
            }
        }
    }

    private fun calculateTip() {

```

```

        val cost =
        costEditText.text.toString().toDoubleOrNull()
        if (cost == null || cost == 0.0) {
            tipResult.text =
            getString(R.string.tip_default) return
        }

        val tipPercentage = when (tipOptions.checkedRadioButtonId) {
            R.id.optionTwenty -> 0.20
            R.id.optionEighteen -> 0.18
            R.id.optionFifteen -> 0.15
            else -> 0.15
        }

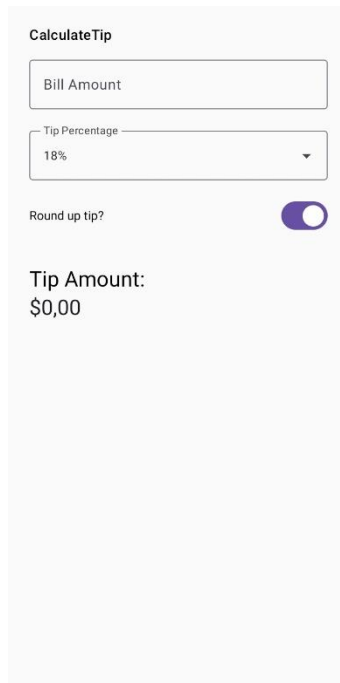
        var tip = cost * tipPercentage if (roundUpSwitch.isChecked) {
            tip = ceil(tip)
        }

        Val formattedTip
        NumberFormat.getCurrencyInstance().format(tip)
        tipResult.text = getString(R.string.tip_amount, formattedTip)
    }
}

```

Tabel 6.Source Code Soal 1 Modul 2

B. Output Program



CalculateTip

Bill Amount

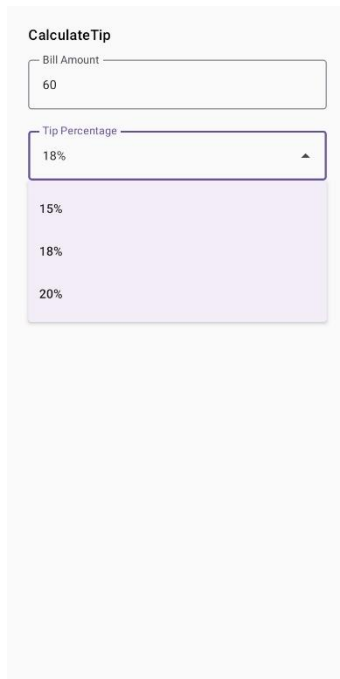
Tip Percentage

18%

Round up tip? ☒

Tip Amount:
\$0,00

Gambar 12. Screenshot Soal 1 Modul 2



CalculateTip

Bill Amount

60

Tip Percentage

18%

15%

18%

20%

Gambar 13. Screenshot Soal 1 Modul 2

Calculate Tip

Bill Amount

60

Tip Percentage

18%

Round up tip? ☒

Tip Amount:

\$11,00

Gambar 14.Screenshot Soal 1 Modul 2

60

☐ 20%

☒ 18%

☐ 15%

Round up tip? ☒

Tip Amount: Rp11,00

CALCULATE

Gambar 15.Screenshot Soal 1 Modul 2

C. Pembahasan

XmlActivity.kt adalah activity berbasis XML layout yang berfungsi sebagai kalkulator tip sederhana. Di sini, komponen UI seperti EditText untuk input biaya layanan, RadioGroup untuk memilih persentase tip, Switch untuk opsi pembulatan tip, dan TextView untuk menampilkan hasil diinisialisasi menggunakan `findViewById`. Kalkulasi tip dilakukan saat pengguna menekan tombol "Done" pada keyboard dengan memanggil fungsi `calculateTip()`, yang mengambil nilai input, menentukan persentase tip berdasarkan RadioButton yang dipilih, menghitung tip, melakukan pembulatan jika diaktifkan, dan menampilkan hasil dalam format mata uang lokal.

activity_xml.xml adalah file layout XML yang mendefinisikan tampilan UI untuk kalkulator tip di aplikasi. Layout ini menggunakan ScrollView dengan LinearLayout vertikal berisi EditText untuk input biaya, RadioGroup dengan tiga RadioButton untuk memilih persentase tip (15%, 18%, 20%), sebuah Switch untuk memilih opsi pembulatan tip, TextView untuk menampilkan hasil perhitungan tip, serta sebuah tombol "Calculate" untuk memicu kalkulasi secara manual. Elemen-elemen diberi padding dan margin agar tampilan lebih rapi dan nyaman digunakan.

MainActivity.kt (XML) adalah activity utama yang menggunakan layout `activity_xml.xml` dan menginisialisasi semua view dengan `findViewById` seperti di XmlActivity, tetapi memiliki tambahan tombol `calculateButton` yang jika diklik akan memicu kalkulasi tip. Selain itu, tombol keyboard "Done" juga di-handle agar memanggil kalkulasi tip. Fungsi `calculateTip()` mirip dengan XmlActivity tapi dengan validasi input tambahan yang memeriksa jika input kosong atau nol, maka menampilkan teks default.

MainActivity.kt (Compose) adalah implementasi kalkulator tip yang sepenuhnya menggunakan Jetpack Compose tanpa layout XML. Aktivitas ini menampilkan UI berbasis composable function yang menggunakan state untuk menangani input biaya, persentase tip yang dipilih dari dropdown menu, dan switch untuk opsi pembulatan. Perhitungan tip dilakukan secara reaktif sesuai state saat ini, hasilnya diformat ke mata uang dolar dan ditampilkan langsung. UI terdiri dari field input teks, dropdown untuk persentase, switch pembulatan, serta tampilan hasil yang diatur dalam Column dengan padding dan spacing agar mudah dibaca dan digunakan.

SOAL 2

Jelaskan perbedaan dari implementasi XML dan Jetpack Compose beserta kelebihan dan kekurangan dari masing-masing implementasi.

Jawabannya :

Perbedaan antara implementasi XML dan Jetpack Compose dalam pengembangan antarmuka pengguna (UI) di Android terletak pada pendekatan dan teknologi yang digunakan. XML merupakan metode deklaratif tradisional yang digunakan untuk mendesain UI di Android. Dalam XML, komponen UI didefinisikan dalam file terpisah dari logika program (biasanya di Activity atau Fragment). Sebaliknya, Jetpack Compose adalah toolkit modern berbasis deklaratif yang memungkinkan pengembang untuk membangun UI langsung di dalam kode Kotlin, sehingga logika dan antarmuka berada dalam satu tempat.

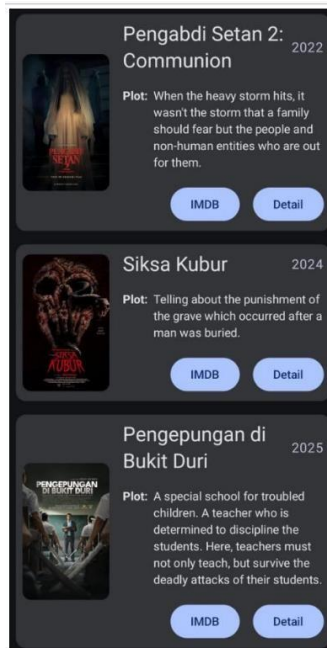
Kelebihan XML adalah stabilitasnya yang telah teruji karena digunakan sejak awal pengembangan Android. XML juga memiliki dukungan luas dari berbagai library dan komunitas. Namun, kekurangannya terletak pada banyaknya *boilerplate code* dan pemisahan antara UI dan logika yang kadang menyulitkan pemeliharaan kode. Sementara itu, Jetpack Compose menawarkan pendekatan yang lebih modern dan efisien, memungkinkan pembuatan UI yang lebih dinamis dan reaktif dengan lebih sedikit kode. Compose juga mendukung pengelolaan *state* secara lebih terintegrasi. Meskipun demikian, kekurangan dari Jetpack Compose adalah kompatibilitasnya dengan sistem lama masih terbatas dan membutuhkan kurva pembelajaran baru bagi pengembang yang terbiasa dengan XML. Dengan demikian, pilihan antara XML dan Jetpack Compose tergantung pada kebutuhan proyek, tim, dan kompatibilitas aplikasi yang sedang dikembangkan.

MODUL 3 : BUILD A SCROLLABLE LIST

SOAL 1

Buatlah sebuah aplikasi Android menggunakan XML dan Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:

1. List menggunakan fungsi RecyclerView (XML) dan LazyColumn (Compose)
2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas
3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah
4. Terdapat 2 button dalam list, dengan fungsi berikut:
 - a. Button pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain
 - b. Button kedua menggunakan Navigation component untuk membuka laman detail item
5. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
6. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang
7. Aplikasi menggunakan arsitektur single activity (satu activity memiliki beberapa fragment)
8. Aplikasi berbasis XML harus menggunakan ViewBinding



Gambar 17. Soal 1 Modul 3

Desain UI laman detail bebas, tetapi diusahakan untuk mengikuti kaidah desain Material Design dan data item ditampilkan penuh di laman detail seperti contoh berikut:



Gambar 18.Soa1 1 Modul 3

A. Source Code

Compose:

MainActivity.kt

```
<!DOCTYPE html>
<html>
<head>
    <title>PRAK301</title>
</head>
<body>
    <form method="post">
        Jumlah Peserta : <input type="number" name="jumlah"
required>
        <input type="submit" value="Cetak">
    </form>
    <br>
    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $jumlah = $_POST["jumlah"];
        $i = 1;
        while ($i <= $jumlah) {
            $warna = ($i % 2 == 1) ? "red" : "green"; echo "<p
style='color:$warna'>Peserta ke-
$i</p>";
            $i++;
        }
    }
}
```

```
?>
</body>
</html>
```

Tabel 7. Source Code Soal 1 Modul 3

FilmData.kt

```
package com.example.listfilm.data

import com.example.listfilm.R

val filmList = listOf(
    Film(
        title = "Happiness",
        description = "Drama Korea Happiness adalah drama Korea yang mengusung tema tentang perjuangan manusia menghadapi penyakit menular di dunia yang terjebak dalam perbedaan kelas sosial.",
        imageResId = R.drawable.happiness,
        url = "https://www.bilibili.tv/id/video/4792916821151744",
        genre = "Action, Thriller, Drama, Sci-Fi",
        year = 2021
    ),
    Film(
        title = "All Of Us Are Dead",
        description = "Sebuah SMA menjadi titik nol merebaknya wabah virus zombi. Para murid yang terperangkap pun harus berjuang untuk kabur jika tak mau terinfeksi dan berubah menjadi buas.",
```



```

        imageResId = R.drawable.aouad,
        url = "https://www.bilibili.tv/id/video/2049830922",
        genre = "Action, Thriller, Horror, Sci-Fi",
        year = 2022
    ),
    Film(
        title = "The Trauma Code",
        description = "Seorang ahli bedah jenius yang tergabung
dengan tim trauma parah memimpin sebuah tim untuk berjuang di
sebuah rumah sakit universitas. ",
        imageResId = R.drawable.traumacenter,
        url = "https://www.bilibili.tv/id/video/4794455678652416",
        genre = "Action, Drama, Medical, Fantasy",
        year = 2025
    ),
    Film(
        title = "Newtopia",
        description = "sebuah drama yang mengisahkan perjalanan
seorang tentara bernama Jae Yoon. Tanpa diduga, ia harus menghadapi
wabah zombie bersama mantan kekasihnya, Kang Yeong Ju.",
        imageResId = R.drawable.newtopia,
        url = "https://www.bilibili.tv/id/video/4794465462128640",
        genre = "Action, Thriller, Romance, Fantasy",
        year = 2025
    ),
    Film(
        title = "Snow Drop",
        description = "Drama Snowdrop ini berlatar Korea Selatan
pada tahun 1987, menceritakan tentang Im Soo Ho (Jung Hae In)
seorang mahasiswa di sebuah universitas bergengsi yang tiba-tiba
masuk ke asrama wanita dengan berlumuran darah. Disana ia bertemu

```

dengan seorang mahasiswi Eun Young Ro (Kim Ji Soo) yang ceria dan menyenangkan. Young Ro lantas menyembunyikan Soo Ho dan merawat lukanya bahkan disaat pengawasan ketat.",

```
        imageResId = R.drawable.snowdrop,  
        url = "https://www.bilibili.tv/id/video/2009361160",  
        genre = "Romance, Drama, Melodrama, Political",  
        year = 2021  
    )  
)
```

Tabel 8.Source Code Soal 1 Modul 3

Film.kt

```
package com.example.listfilm.data  
  
data class Film(  
    val title: String,  
    val year: Int,  
    val genre: String,  
    val description: String,  
    val imageResId: Int,  
    val url: String  
)
```

Tabel 9.Source Code Soal 1 Modul 3

FilmDetailScreen.kt

```
package com.example.listfilm.ui.screen  
  
import androidx.compose.foundation.Image  
import androidx.compose.foundation.layout.*  
import  
    androidx.compose.foundation.shape.RoundedCornerShape  
import androidx.compose.material3.MaterialTheme
```

```

import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import com.example.listfilm.data.Film

@Composable
fun FilmDetailScreen(film: Film) {
    Column(modifier = Modifier.padding(16.dp)) {
        Image(
            painter = painterResource(id = film.imageResId),
            contentDescription = null,
            modifier = Modifier
                .fillMaxWidth()
                .height(200.dp)
                .clip(RoundedCornerShape(12.dp)),
            contentScale = ContentScale.Crop
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(text = film.title, style =
MaterialTheme.typography.headlineSmall)
        Text(text = "${film.genre} • ${film.year}")
        Spacer(modifier = Modifier.height(8.dp))
        Text(text = film.description)
    }
}

```

Tabel 10.Source Code Soal 1 Modul 3

FilmListScreen.kt

```

package com.example.listfilm.ui.screen

import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn

```

```

import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.Card
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.core.net.toUri
import androidx.navigation.NavHostController
import com.example.listfilm.viewmodel.FilmViewModel

@Composable
fun FilmListScreen(navController: NavHostController, viewModel:
FilmViewModel) {
    val context = LocalContext.current
    val films = viewModel.films.collectAsState().value

    LazyColumn(contentPadding = PaddingValues(16.dp)) {
        items(films) { film ->
            Card(
                shape = RoundedCornerShape(16.dp),
                modifier = Modifier
                    .padding(bottom = 16.dp)
                    .fillMaxWidth()
            ) {
                Column(Modifier.padding(16.dp)) {
                    Image(
                        painter = painterResource(id = film.imageResId),
                        contentDescription = null,
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp)
                            .clip(RoundedCornerShape(12.dp)),
                        contentScale = ContentScale.Crop
                    )
                    Spacer(Modifier.height(8.dp))
                    Row(
                        horizontalArrangement =
Arrangement.SpaceBetween,
                        modifier = Modifier.fillMaxWidth()

```


AppNavGraph.kt

```
package com.example.listfilm.ui.screen

import android.net.Uri
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.ui.Modifier
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import com.example.listfilm.viewmodel.FilmViewModel

@Composable
fun AppNavGraph(navController: NavHostController, viewModel: FilmViewModel) {
    val films = viewModel.films.collectAsState().value
    NavHost(navController = navController, startDestination = "list") {
        composable("list") {
            FilmListScreen(navController = navController, viewModel = viewModel)
        }
        composable("detail/{title}") { backStackEntry ->
            val rawTitle = backStackEntry.arguments?.getString("title")
            val decodedTitle = rawTitle?.let { Uri.decode(it) }
            val film = films.find { it.title == decodedTitle }
            if (film != null) {
                FilmDetailScreen(film = film)
            } else {
                Text("Film not found", modifier = Modifier.fillMaxSize())
            }
        }
    }
}
```

Tabel 12.Source Code Soal 1 Modul 3

XML

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
```

```

xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent">

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:padding="16dp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>

```

Tabel 13.Source Code Soal 1 Modul 3

activity_detail.xml

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:scaleType="centerCrop" />

        <TextView
            android:id="@+id/titleText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:textStyle="bold"
            android:layout_marginTop="12dp" />
    </LinearLayout>
</ScrollView>

```

```

        <TextView
            android:id="@+id/detailText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp" />
    </LinearLayout>
</ScrollView>

```

Tabel 14.Source Code Soal 1 Modul 3

item_film.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginTop="8dp"
    android:elevation="4dp"
    android:padding="12dp"
    android:radius="12dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:scaleType="centerCrop" />

        <TextView
            android:id="@+id/titleText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="16sp"
            android:textStyle="bold"
            android:paddingTop="8dp"/>

        <TextView
            android:id="@+id/yearGenreText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

```



```

        android:textColor="#888888"/>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginTop="8dp"
            android:gravity="end">

            <Button
                android:id="@+id/detailButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Detail" />

            <Button
                android:id="@+id/siteButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Open Site"
                android:layout_marginStart="8dp"/>
        </LinearLayout>
    </LinearLayout>
</androidx.cardview.widget.CardView>

```

Tabel 15.Source Code Soal 1 Modul 3

DetailActivity.kt

```

package com.example.listfilm

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import com.example.listfilm.data.filmList
import com.example.listfilm.databinding.ActivityDetailBinding

class DetailActivity : AppCompatActivity() {
    private lateinit var binding: ActivityDetailBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityDetailBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val title = intent.getStringExtra("title")
        val film = filmList.find { it.title == title }
    }
}

```

```

        film?.let {
            binding.imageView.setImageResource(it.imageResId)
            binding.titleText.text = it.title
            binding.detailText.text = "${it.genre} •
${it.year}\n\n${it.description}"
        } ?: run {
            binding.titleText.text = "Film not found"
        }
    }
}

```

Tabel 16.Source Code Soal 1 Modul 3

Film.kt

```

package com.example.listfilm.data

data class Film(
    val title: String,
    val genre: String,
    val year: Int,
    val description: String,
    val imageResId: Int,
    val url: String
)

```

Tabel 17.Source Code Soal 1 Modul 3

FilmAdapter.kt

```

package com.example.listfilm

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.listfilm.data.Film
import com.example.listfilm.databinding.ItemFilmBinding

class FilmAdapter(
    private val filmList: List<Film>,
    private val onDetailClick: (Film) -> Unit,
    private val onOpenSiteClick: (Film) -> Unit
) : RecyclerView.Adapter<FilmAdapter.FilmViewHolder>() {

```

```

        inner class FilmViewHolder(private val binding:
ItemFilmBinding) :
    RecyclerView.ViewHolder(binding.root) {
        fun bind(film: Film) {
            binding.imageView.setImageResource(film.imageResId)
            binding.titleText.text = film.title
            binding.yearGenreText.text = "${film.year} •
${film.genre}"
            binding.detailButton.setOnClickListener {
onDetailClick(film) }
            binding.siteButton.setOnClickListener {
onOpenSiteClick(film) }
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): FilmViewHolder {
        val binding =
ItemFilmBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
        return FilmViewHolder(binding)
    }

    override fun onBindViewHolder(holder: FilmViewHolder, position:
Int) {
        holder.bind(filmList[position])
    }

    override fun getItemCount(): Int = filmList.size
}

```

Tabel 18.Source Code Soal 1 Modul 3

MainActivity.kt

```

package com.example.listfilm

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.listfilm.data.filmList
import com.example.listfilm.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

```

```

private lateinit var binding: ActivityMainBinding

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

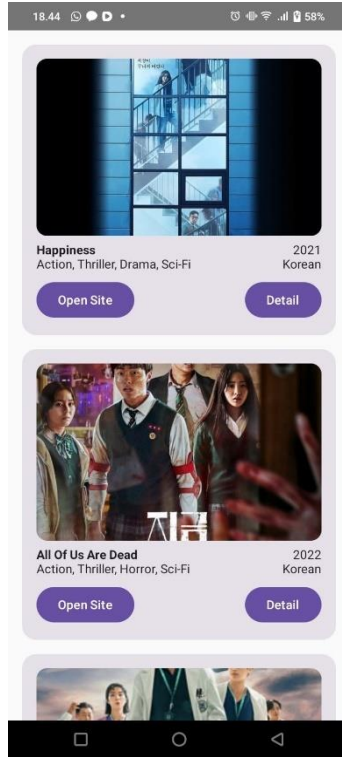
    val adapter = FilmAdapter(
        filmList,
        onDetailClick = { film ->
                                val intent = Intent(this,
DetailActivity::class.java)
                                intent.putExtra("title", film.title)
                                startActivity(intent)
        },
        onOpenSiteClick = { film ->
                                val intent = Intent(Intent.ACTION_VIEW,
Uri.parse(film.url))
                                startActivity(intent)
        }
    )

    binding.recyclerView.layoutManager =
LinearLayoutManager(this)
    binding.recyclerView.adapter = adapter
}
}

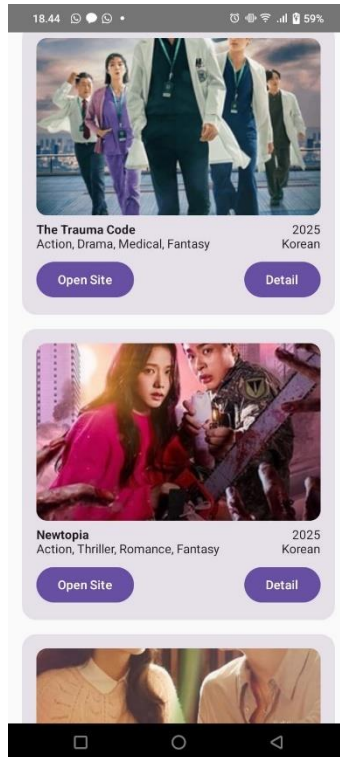
```

Tabel 19.Source Code Soal 1 Modul 3

B. Output Program



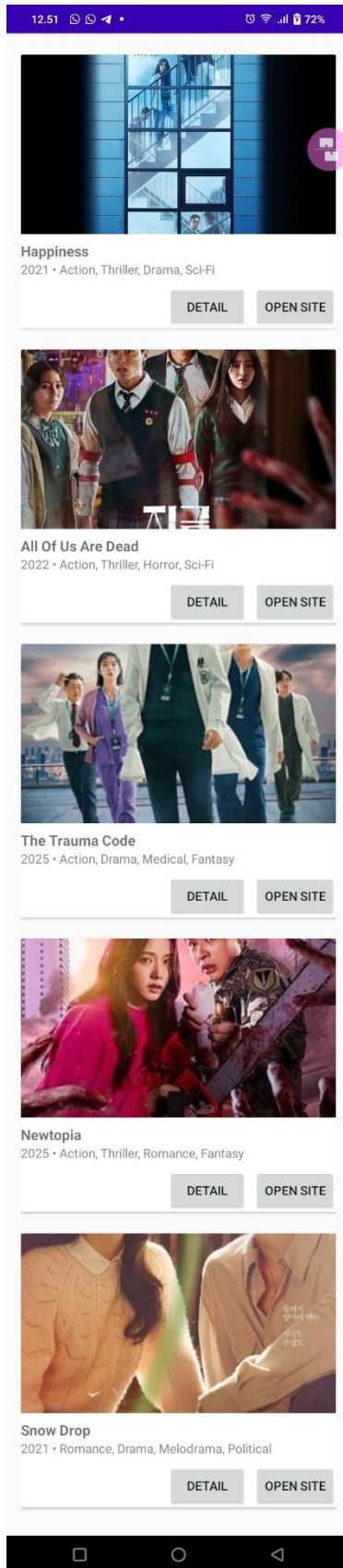
Gambar 19.Screenshot Jawaban Hasil Soal 1 Modul 3



Gambar 4. Soal 1 (Compose)



Gambar 20. Screenshot Jawaban Hasil Soal 1 Modul 3



Gambar 21. Screenshot Jawaban Hasil Soal 1 Modul 3



Gambar 22. Screenshot Hasil Jawaban Soal 1 Modul 3

C. Pembahasan

Kode di atas merupakan implementasi aplikasi daftar film yang dibangun dengan dua pendekatan antarmuka pengguna di Android: **Jetpack Compose** dan **XML tradisional**. Aplikasi ini memungkinkan pengguna untuk melihat daftar film, melihat detail setiap film, dan membuka situs eksternal terkait film tersebut. Data film disimpan dalam `FilmData.kt` dalam bentuk list `filmList`, yang masing-masing item-nya adalah objek dari kelas `Film`, berisi informasi seperti judul, deskripsi, genre, tahun rilis, URL, dan ID gambar.

Untuk pendekatan **Jetpack Compose**, tampilan utama ditangani oleh `FilmListScreen.kt`, yang menggunakan `LazyColumn` untuk menampilkan daftar film secara efisien. Setiap item ditampilkan dalam komponen `Card`, yang berisi gambar, judul, dan tahun rilis film. Ketika item dipilih, pengguna dapat dinavigasi ke halaman detail melalui `NavController`. Tampilan detail film dikelola oleh `FilmDetailScreen.kt`, yang menampilkan gambar besar, judul, genre dan tahun, serta deskripsi film menggunakan elemen-elemen `Compose` seperti `Image`, `Text`, dan `Spacer`. Navigasi antar layar diatur oleh `NavGraph.kt`, meskipun kode lengkapnya tidak ditampilkan.

Sementara itu, dalam pendekatan **XML konvensional**, file seperti `activity_main.xml`, `activity_detail.xml`, dan `item_film.xml` digunakan untuk mendefinisikan tampilan secara statis. Komponen seperti `RecyclerView` digunakan untuk menampilkan daftar film, dan `FilmAdapter.kt` bertanggung jawab atas pengisian data ke dalam item tampilan. `FilmAdapter` mengatur `ViewHolder`, dan pada fungsi `onBindViewHolder`, gambar, judul, dan info film ditampilkan, serta dua tombol diatur: satu untuk membuka link ke situs film menggunakan `Intent.ACTION_VIEW`, dan satu lagi untuk membuka `DetailActivity` yang menampilkan detail film.

Dengan dua pendekatan ini, pengguna dapat menelusuri daftar film, membuka detail, serta menavigasi ke tautan eksternal untuk menonton atau mengetahui lebih lanjut. Jetpack Compose menawarkan pendekatan deklaratif modern, sementara XML dan `RecyclerView` masih digunakan untuk kompatibilitas dan kontrol langsung terhadap UI. Kombinasi data model (`Film`), pengelolaan UI, dan navigasi membentuk aplikasi yang utuh dan interaktif.

SOAL 2

Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

Jawabannya :

Menurut saya, alasan kenapa RecyclerView masih digunakan sampai sekarang, meskipun kodenya terkesan panjang dan boiler-plate dibandingkan dengan LazyColumn di Jetpack Compose, ada beberapa poin penting.

Pertama, RecyclerView masih sangat kompatibel dengan View system lama di Android. Banyak aplikasi yang masih menggunakan XML dan belum sepenuhnya migrasi ke Compose, jadi RecyclerView tetap jadi pilihan utama karena bisa langsung diintegrasikan tanpa harus mengubah struktur besar-besaran.

Kedua, RecyclerView menawarkan fleksibilitas dan kontrol yang sangat tinggi. Saya bisa menggunakan LayoutManager custom, animasi transisi item lewat ItemAnimator, atau bahkan pakai SnapHelper. Ini penting banget kalau butuh UI yang kompleks.

Selain itu, karena RecyclerView sudah ada sejak lama, dokumentasinya lengkap, tutorialnya banyak, dan komunitasnya luas. Kalau ada masalah, saya bisa dengan mudah cari solusi di Stack Overflow atau artikel lainnya.

Jujur, meskipun LazyColumn di Compose jauh lebih ringkas dan efisien untuk use case sederhana, tapi Compose sendiri masih terus berkembang dan belum bisa menggantikan semua fitur lanjutan yang dimiliki RecyclerView. Terutama kalau butuh nested scroll yang kompleks atau integrasi dengan komponen View yang lebih lama.

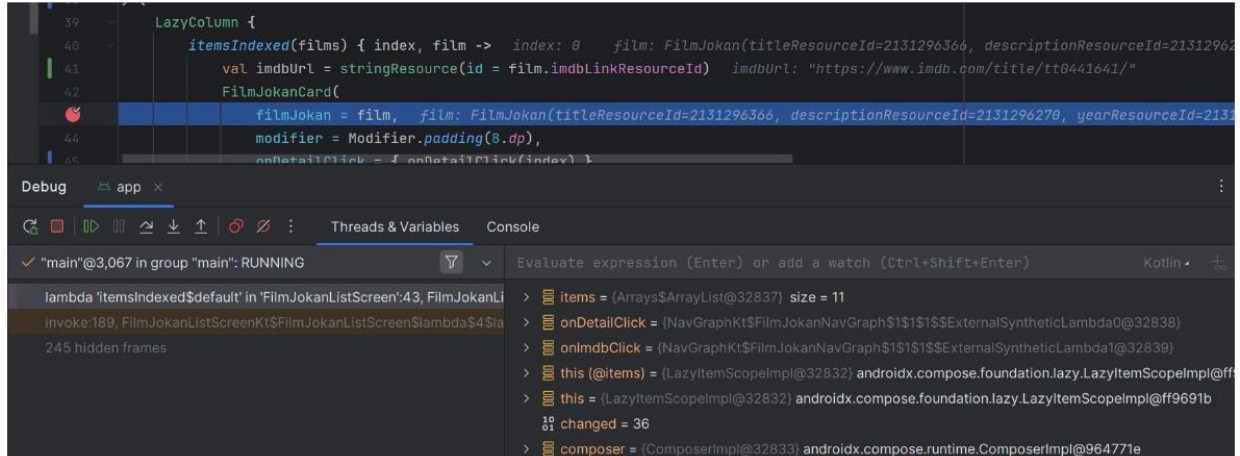
Jadi menurut saya, RecyclerView masih sangat relevan, khususnya untuk proyek-proyek besar atau aplikasi yang belum full Compose. Tapi kalau saya bikin aplikasi baru dan pengen UI modern yang lebih cepat dikembangkan, tentu saya bakal pilih LazyColumn.

MODUL 4 : VIEW MODEL DAN DEBUGGING

SOAL 1

Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:

- Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
- Gunakan ViewModelFactory untuk membuat parameter dengan tipe data String di dalam ViewModel
- Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
- Install dan gunakan library Timber untuk logging event berikut:
 - Log saat data item masuk ke dalam list
 - Log saat tombol Detail dan tombol Explicit Intent ditekan
 - Log data dari list yang dipilih ketika berpindah ke halaman Detail
- Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi. Cari setidaknya satu breakpoint yang relevan dengan aplikasi. Lalu, gunakan fitur Step Into, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out



Gambar 23. Soal 1 Modul 4

A. Source Code

Compose

MainActivity.kt

```

<!DOCTYPE html>
<html>
<head>
    <title>PRAK301</title>
</head>
<body>
    <form method="post">
        Jumlah Peserta : <input type="number" name="jumlah"
required>
        <input type="submit" value="Cetak">
    </form>
    <br>
    <?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $jumlah = $_POST["jumlah"];
        $i = 1;
        while ($i <= $jumlah) {
            $warna = ($i % 2 == 1) ? "red" : "green"; echo "<p
style='color:$warna'>Peserta ke-
$i</p>";
            $i++;
        }
    }
    ?>
</body>
</html>

```

Tabel 20.Source Code Soal 1 Modul 4

FilmData.kt

```
package com.example.listfilm.data
```

```

import com.example.listfilm.R

val filmList = listOf(
    Film(
        title = "Happiness",
        description = "Drama Korea Happiness adalah drama Korea yang mengusung tema tentang perjuangan manusia menghadapi penyakit menular di dunia yang terjebak dalam perbedaan kelas sosial.",
        imageResId = R.drawable.happiness,
        url = "https://www.bilibili.tv/id/video/4792916821151744",
        genre = "Action, Thriller, Drama, Sci-Fi",
        year = 2021
    ),
    Film(
        title = "All Of Us Are Dead",
        description = "Sebuah SMA menjadi titik nol merebaknya wabah virus zombi. Para murid yang terperangkap pun harus berjuang untuk kabur jika tak mau terinfeksi dan berubah menjadi buas.",
        imageResId = R.drawable.aouad,
        url = "https://www.bilibili.tv/id/video/2049830922",
        genre = "Action, Thriller, Horror, Sci-Fi",
        year = 2022
    ),
    Film(
        title = "The Trauma Code",
        description = "Seorang ahli bedah jenius yang tergabung dengan tim trauma parah memimpin sebuah tim untuk berjuang di sebuah rumah sakit universitas. ",
        imageResId = R.drawable.traumacenter,
        url = "https://www.bilibili.tv/id/video/4794455678652416",
        genre = "Action, Drama, Medical, Fantasy",
    )
)

```

```

        year = 2025
    ),
    Film(
        title = "Newtopia",
        description = "sebuah drama yang mengisahkan perjalanan
seorang tentara bernama Jae Yoon. Tanpa diduga, ia harus menghadapi
wabah zombie bersama mantan kekasihnya, Kang Yeong Ju.",
        imageResId = R.drawable.newtopia,
        url = "https://www.bilibili.tv/id/video/4794465462128640",
        genre = "Action, Thriller, Romance, Fantasy",
        year = 2025
    ),
    Film(
        title = "Snow Drop",
        description = "Drama Snowdrop ini berlatar Korea Selatan
pada tahun 1987, menceritakan tentang Im Soo Ho (Jung Hae In)
seorang mahasiswa di sebuah universitas bergengsi yang tiba-tiba
masuk ke asrama wanita dengan berlumuran darah. Disana ia bertemu
dengan seorang mahasiswi Eun Young Ro (Kim Ji Soo) yang ceria dan
menyenangkan. Young Ro lantas menyembunyikan Soo Ho dan merawat
lukanya bahkan disaat pengawasan ketat.",
        imageResId = R.drawable.snowdrop,
        url = "https://www.bilibili.tv/id/video/2009361160",
        genre = "Romance, Drama, Melodrama, Political",
        year = 2021
    )
)

```

Tabel 21.Source Code Soal 1 Modul 4

Film.kt

```
package com.example.listfilm.data

data class Film(
    val title: String,
    val year: Int,
    val genre: String,
    val description: String,
    val imageResId: Int,
    val url: String
)
```

Tabel 22.Source Code Soal 1 Modul 4

FilmDetailScreen.kt

```
package com.example.listfilm.ui.screen

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import com.example.listfilm.data.Film

@Composable
fun FilmDetailScreen(film: Film) {
    Column(modifier = Modifier.padding(16.dp)) {
        Image(
            painter = painterResource(id = film.imageResId),
            contentDescription = null,
            modifier = Modifier
                .fillMaxWidth()
                .height(200.dp)
                .clip(RoundedCornerShape(12.dp)),
            contentScale = ContentScale.Crop
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(text = film.title, style =
MaterialTheme.typography.headlineSmall)
```

```

        Text(text = "${film.genre} • ${film.year}")
        Spacer(modifier = Modifier.height(8.dp))
        Text(text = film.description)
    }
}

```

Tabel 23.Source Code Soal 1 Modul 4

FilmListScreen.kt

```

package com.example.listfilm.ui.screen

import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.Card
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.core.net.toUri
import androidx.navigation.NavHostController
import com.example.listfilm.viewmodel.FilmViewModel

@Composable
fun FilmListScreen(navController: NavHostController, viewModel:
FilmViewModel) {
    val context = LocalContext.current
    val films = viewModel.films.collectAsState().value

```



```

LazyColumn(contentPadding = PaddingValues(16.dp)) {
    items(films) { film ->
        Card(
            shape = RoundedCornerShape(16.dp),
            modifier = Modifier
                .padding(bottom = 16.dp)
                .fillMaxWidth()
        ) {
            Column(Modifier.padding(16.dp)) {
                Image(
                    painter = painterResource(id =
film.imageResId),
                    contentDescription = null,
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(200.dp)
                        .clip(RoundedCornerShape(12.dp)),
                    contentScale = ContentScale.Crop
                )
                Spacer(Modifier.height(8.dp))
                Row(
                    horizontalArrangement =
Arrangement.SpaceBetween,
                    modifier = Modifier.fillMaxWidth()
                ) {
                    Text(text = film.title, fontWeight =
FontWeight.Bold)
                    Text(text = "${film.year}")
                }
                Row(
                    horizontalArrangement =
Arrangement.SpaceBetween,
                    modifier = Modifier.fillMaxWidth()
                ) {
                    Text(text = film.genre)
                    Text(text = "Korean")
                }
                Spacer(Modifier.height(8.dp))
                Row(
                    modifier = Modifier.fillMaxWidth(),
                    horizontalArrangement =
Arrangement.SpaceBetween
                ) {
                    Button(onClick = {
                        viewModel.logOpenSiteClicked(film)
                    })
                }
            }
        }
    }
}

```



```

                                val rawTitle =
backStackEntry.arguments?.getString("title")
                                val decodedTitle = rawTitle?.let { Uri.decode(it) }
                                val film = films.find { it.title == decodedTitle }
                                if (film != null) {
                                    FilmDetailScreen(film = film)
                                } else {
                                    Text("Film not found", modifier =
Modifier.fillMaxSize())
                                }
                            }
                        }
                    }
}

```

Tabel 25.Source Code Soal 1 Modul 4

FilmViewModel.kt

```

package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.listfilm.data.Film
import com.example.listfilm.data.filmList
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.launch
import timber.log.Timber

class FilmViewModel(private val source: String) : ViewModel() {
    private val _films = MutableStateFlow<List<Film>>(emptyList())
    val films: StateFlow<List<Film>> = _films

    init {
        viewModelScope.launch {
            _films.value = filmList
            Timber.d("Film list loaded from $source with
${filmList.size} items")
        }
    }

    fun logDetailClicked(film: Film) {
        Timber.d("Detail clicked: ${film.title}")
    }

    fun logOpenSiteClicked(film: Film) {
        Timber.d("Open site clicked: ${film.url}")
    }
}

```

```
}
}
```

Tabel 26.Source Code Soal 1 Modul 4

FilmViewModelFactory.kt

```
package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class FilmViewModelFactory(private val source: String) :
    ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if
        (modelClass.isAssignableFrom(FilmViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
            return FilmViewModel(source) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

Tabel 27.Source Code Soal 1 Modul 4

XML

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:padding="16dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 28.Source Code Soal 1 Modul 4

activity_detail.xml

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:scaleType="centerCrop" />

        <TextView
            android:id="@+id/titleText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:textStyle="bold"
            android:layout_marginTop="12dp" />

        <TextView
            android:id="@+id/detailText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp" />
    </LinearLayout>
</ScrollView>

```

Tabel 29.Source Code Soal 1 Modul 4

item_film.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="16dp"
    android:layout_marginTop="8dp"
    android:elevation="4dp"
    android:padding="12dp"

```

```

        android:radius="12dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <ImageView
                android:id="@+id/imageView"
                android:layout_width="match_parent"
                android:layout_height="200dp"
                android:scaleType="centerCrop" />

            <TextView
                android:id="@+id/titleText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="16sp"
                android:textStyle="bold"
                android:paddingTop="8dp"/>

            <TextView
                android:id="@+id/yearGenreText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textColor="#888888"/>

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:layout_marginTop="8dp"
                android:gravity="end">

                <Button
                    android:id="@+id/detailButton"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Detail" />

                <Button
                    android:id="@+id/siteButton"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="Open Site"
                    android:layout_marginStart="8dp"/>
            </LinearLayout>

```

```
</LinearLayout>
</androidx.cardview.widget.CardView>
```

Tabel 30.Source Code Soal 1 Modul 4

DetailActivity.kt

```
package com.example.listfilm

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import com.example.listfilm.data.filmList
import com.example.listfilm.databinding.ActivityDetailBinding

class DetailActivity : AppCompatActivity() {
    private lateinit var binding: ActivityDetailBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityDetailBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val title = intent.getStringExtra("title")
        val film = filmList.find { it.title == title }

        film?.let {
            binding.imageView.setImageResource(it.imageResId)
            binding.titleText.text = it.title
            binding.detailText.text = "${it.genre} •
            ${it.year}\n\n${it.description}"
        } ?: run {
            binding.titleText.text = "Film not found"
        }
    }
}
```

Tabel 31.Source Code Soal 1 Modul 4

Film.kt

```
package com.example.listfilm.data

data class Film(
    val title: String,
    val genre: String,
    val year: Int,
    val description: String,
```

```

        val imageResId: Int,
        val url: String
    )

```

Tabel 32.Source Code Soal 1 Modul 4

FilmAdapter.kt

```

package com.example.listfilm

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.example.listfilm.data.Film
import com.example.listfilm.databinding.ItemFilmBinding

class FilmAdapter(
    private val filmList: List<Film>,
    private val onDetailClick: (Film) -> Unit,
    private val onOpenSiteClick: (Film) -> Unit
) : RecyclerView.Adapter<FilmAdapter.FilmViewHolder>() {

    inner class FilmViewHolder(private val binding:
ItemFilmBinding) :
        RecyclerView.ViewHolder(binding.root) {
        fun bind(film: Film) {
            binding.imageView.setImageResource(film.imageResId)
            binding.titleText.text = film.title
            binding.yearGenreText.text = "${film.year} •
${film.genre}"
            binding.detailButton.setOnClickListener {
onDetailClick(film) }
            binding.siteButton.setOnClickListener {
onOpenSiteClick(film) }
        }
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): FilmViewHolder {
        val binding =
ItemFilmBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
        return FilmViewHolder(binding)
    }
}

```



```

        override fun onBindViewHolder(holder: FilmViewHolder, position:
Int) {
            holder.bind(filmList[position])
        }

        override fun getItemCount(): Int = filmList.size
    }

```

Tabel 33.Source Code Soal 1 Modul 4

MainActivity.kt

```

package com.example.listfilm

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.listfilm.data.filmList
import com.example.listfilm.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val adapter = FilmAdapter(
            filmList,
            onDetailClick = { film ->
                val intent = Intent(this,
DetailActivity::class.java)
                intent.putExtra("title", film.title)
                startActivity(intent)
            },
            onOpenSiteClick = { film ->
                val intent = Intent(Intent.ACTION_VIEW,
Uri.parse(film.url))
                startActivity(intent)
            }
        )
    }
}

```

```

binding.recyclerView.layoutManager =
LinearLayoutManager(this)
binding.recyclerView.adapter = adapter
}
}

```

Tabel 34.Source Code Soal 1 Modul 4

FilmViewModel.kt

```

package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import com.example.listfilm.data.Film
import com.example.listfilm.data.filmList
import timber.log.Timber

class FilmViewModel : ViewModel() {

    private val _films = MutableStateFlow<List<Film>>(emptyList())
    val films: StateFlow<List<Film>> = _films

    init {
        _films.value = filmList
        Timber.i("Film list dimuat: ${_films.value.size} item")
    }

    fun selectFilm(film: Film) {
        Timber.i("Film dipilih: ${film.title}")
    }
}

```

Tabel 35.Source Code Soal 1 Modul 4

FilmViewModelFactory.kt

```

package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

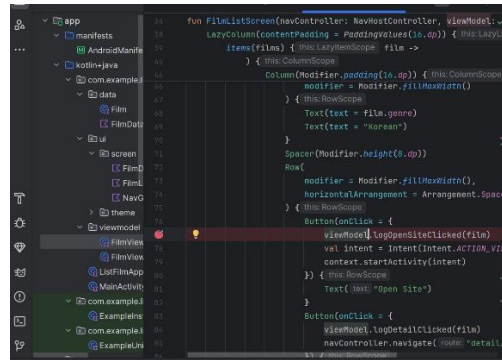
class FilmViewModelFactory(private val param: String) :
    ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if
        (modelClass.isAssignableFrom(FilmViewModel::class.java)) {
            return FilmViewModel() as T
        }
    }
}

```

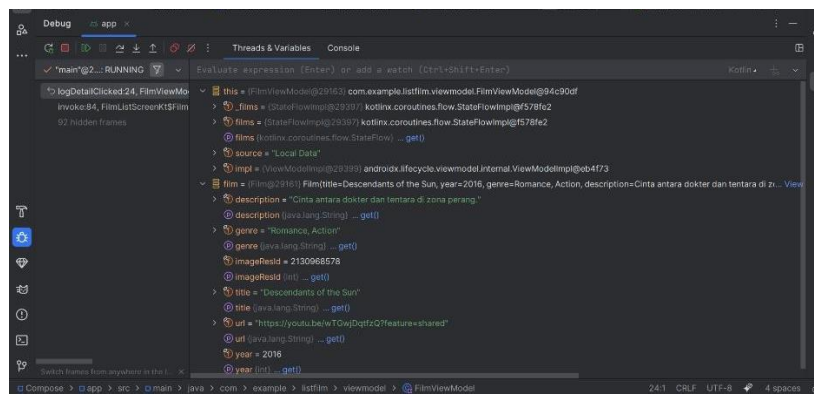
```
        }  
        throw IllegalArgumentException("Unknown ViewModel class")  
    }  
}
```

Tabel 36.Source Code Soal 1 Modul 4

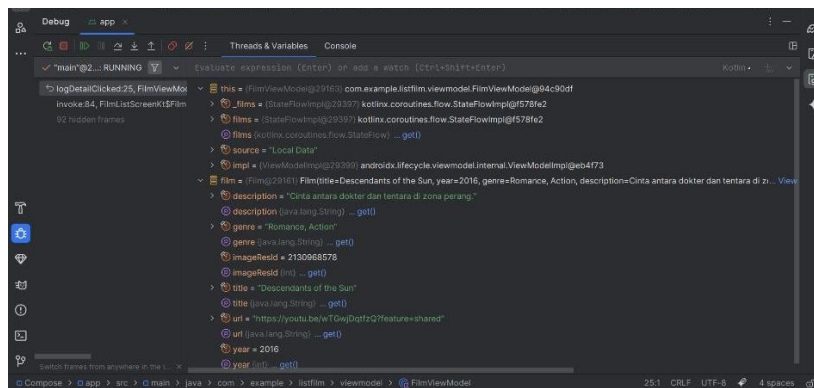
B. Output Program



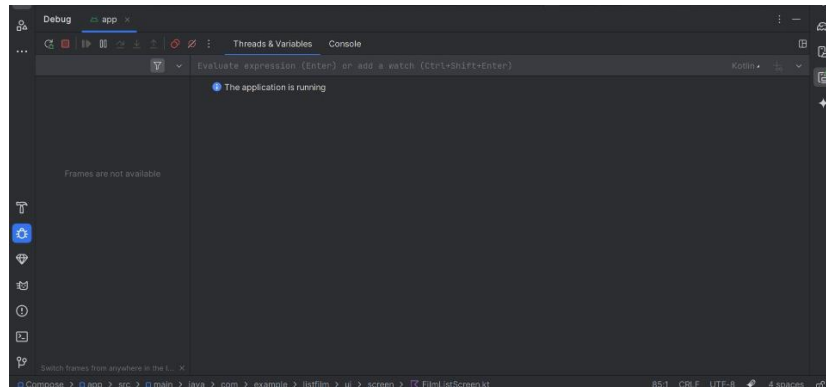
Gambar 24. Breakpoint (Compose)



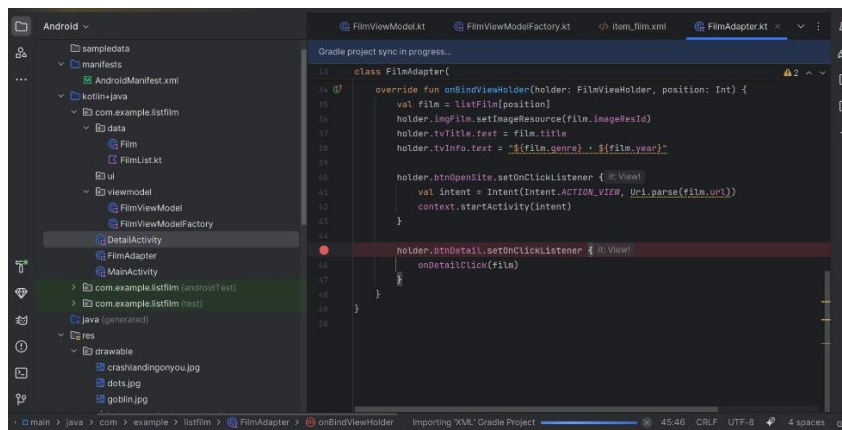
Gambar 25. Step Over (F8) Digunakan (Compose)



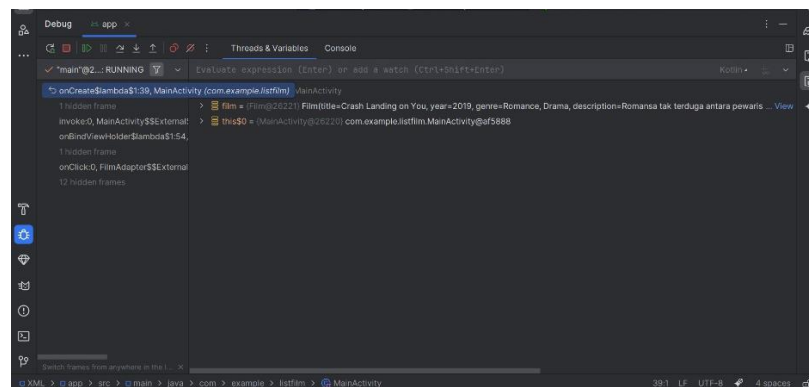
Gambar 26. Step Out (Shift + F8) Keluar Dari Fungsi (Compose)



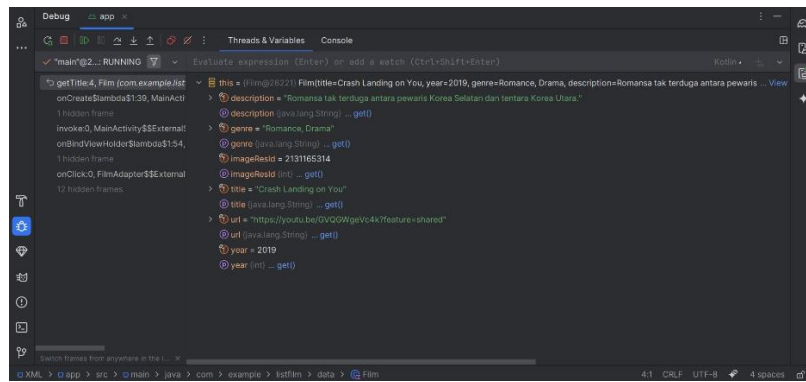
Gambar 27. Resume (F9) Untuk Melanjutkan Eksekusi (Compose)



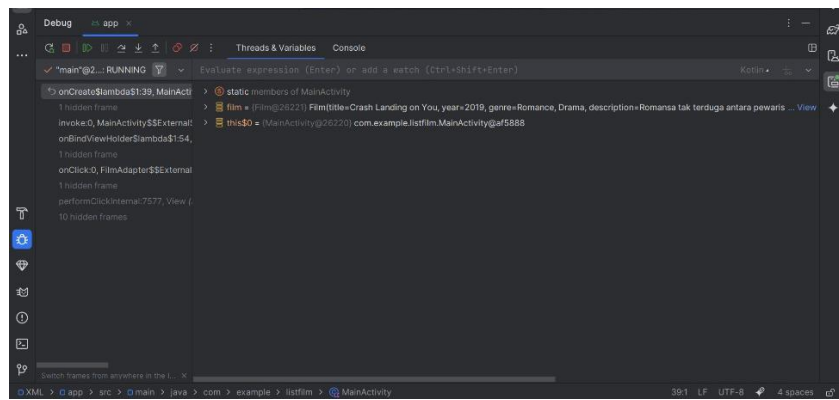
Gambar 28. Breakpoint (XML)



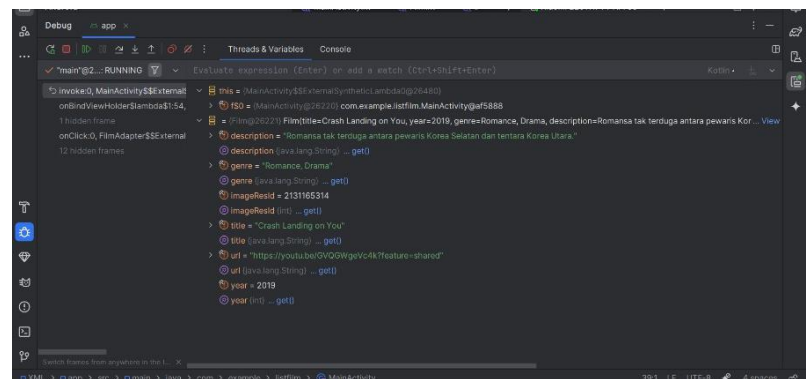
Gambar 29. Debug (XML)



Gambar 30. Step Into (F7) Masuk Ke Fungsi (XML)



Gambar 31. Step Over (F8) Digunakan (XML)



Gambar 32. Step Out (Shift + F8) Keluar Dari Fungsi (XML)

C. Pembahasan

Kode ini lanjutan dari Modul 3 dengan penambahan beberapa modifikasi penting sesuai instruksi. Pertama, untuk memenuhi poin **(a)**, digunakan kelas `FilmViewModel` sebagai `ViewModel` yang bertanggung jawab penuh terhadap pengelolaan dan penyimpanan data list item (film). Dengan pendekatan ini, data tidak disimpan di `Activity` atau `Fragment`, melainkan dikelola secara terpusat dan reaktif melalui `ViewModel`, yang memastikan arsitektur aplikasi bersih dan mengikuti prinsip *separation of concerns*.

Pada poin **(b)**, implementasi `FilmViewModelFactory` digunakan untuk memberikan parameter bertipe `String` (dalam hal ini, nama pengguna seperti "Devi") ke dalam `ViewModel`. `Factory` ini memungkinkan `ViewModelProvider` membuat instance `ViewModel` dengan parameter tambahan, karena secara default `ViewModel` tidak menerima argumen konstruktor.

Untuk poin **(c)**, aplikasi ini menggunakan `StateFlow` pada `FilmViewModel` untuk mengelola data daftar film (`_filmList`) dan film yang dipilih (`_selectedFilm`). Dengan menggunakan `StateFlow`, UI (seperti `RecyclerView` di XML dan `LazyColumn` di `Compose`) dapat berlangganan pada data dan secara otomatis *recompose* atau *refresh* ketika ada perubahan. Selain itu, aksi `onClick` pada tombol "Detail" juga dikirim ke `ViewModel` menggunakan fungsi `selectFilm(film)` yang memperbarui `_selectedFilm`.

Pada poin **(d)**, logging menggunakan **Timber** telah diimplementasikan. Logging ini dilakukan saat:

- Data film pertama kali dimuat dari `filmList` (`loadFilms()`), dengan mencetak jumlah item.
- Tombol "Detail" ditekan pada setiap item (baik di `Compose` maupun XML/`RecyclerView`), yang mencetak judul film yang diklik.
- Saat berpindah ke halaman `DetailActivity`, data film yang dipilih juga dicatat menggunakan `Timber.i`.

Untuk poin **(e)**, proses debugging dilakukan menggunakan **Debugger** di `Android Studio`. Sebuah *breakpoint* yang relevan dapat ditempatkan pada fungsi `onBindViewHolder` di `FilmAdapter`, tepat saat tombol "Detail" ditekan. Dari sini, fitur:

- **Step Into** digunakan untuk masuk ke dalam fungsi `onDetailClick()` untuk melihat bagaimana data dikirim ke `Intent`.
- **Step Over** digunakan untuk mengeksekusi baris kode saat ini dan melanjutkan ke baris berikutnya tanpa masuk ke dalam fungsi yang dipanggil.
- **Step Out** digunakan untuk keluar dari fungsi yang sedang dieksekusi dan kembali ke fungsi pemanggil sebelumnya.

`Debugger` sangat bermanfaat dalam mengidentifikasi alur eksekusi kode dan mencari tahu nilai variabel secara langsung saat aplikasi berjalan. Dengan memanfaatkan fitur-fitur tersebut, pengembang dapat melacak perilaku aplikasi dengan lebih akurat, terutama ketika terjadi bug atau error.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya!

Jawabannya:

Application class dalam arsitektur aplikasi Android adalah kelas dasar yang digunakan untuk menyimpan *state global* aplikasi. Kelas ini merupakan turunan dari kelas `android.app.Application` dan berperan sebagai titik awal ketika aplikasi pertama kali dijalankan, sebelum Activity, Service, atau komponen lain dibuat.

Fungsi utama dari **Application class** adalah untuk menginisialisasi hal-hal yang hanya perlu dilakukan sekali selama siklus hidup aplikasi. Misalnya, mengatur dependency injection (seperti Hilt atau Dagger), menginisialisasi pustaka pihak ketiga (seperti Firebase, Retrofit, atau Timber), menyimpan konfigurasi global, serta menyediakan konteks aplikasi secara global melalui `getApplicationContext()`.

Dengan mendefinisikan subclass dari Application dan mendeklarasikannya di file `AndroidManifest.xml`, pengembang bisa mengontrol perilaku global aplikasi secara terpusat. Ini sangat membantu dalam membangun aplikasi yang terstruktur dengan baik, terutama yang menggunakan arsitektur modular atau berbasis komponen.

Singkatnya, **Application class** adalah tempat terbaik untuk melakukan konfigurasi awal yang berlaku untuk seluruh aplikasi Android.

MODUL 5 : FUNCTION DAN DATABASE

SOAL 1

Lanjutkan aplikasi Android yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi sesuai ketentuan berikut:

a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.

b. Gunakan KotlinX Serialization sebagai library JSON.

c. Gunakan library seperti Coil atau Glide untuk image loading. d. API yang digunakan pada modul ini adalah The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: <https://developer.themoviedb.org/docs/getting-started>

e. Implementasikan konsep data persistence (aplikasi menyimpan data walau pengguna keluar dari aplikasi) dengan SharedPreferences untuk menyimpan data ringan (seperti pengaturan aplikasi) dan Room untuk data relasional.

f. Gunakan caching strategy pada Room. Dibebaskan untuk memilih caching strategy yang sesuai, dan sertakan penjelasan kenapa menggunakan caching strategy tersebut.

g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose. Aplikasi harus mempertahankan fitur-fitur yang dibuat pada modul sebelumnya

Buatlah sebuah aplikasi berbasis web sederhana yang dapat melakukan operasi CRUD (Create, Read, Update, Delete) dari hasil implementasi desain basis data yang diberikan. Adapun ketentuan pembuatannya sebagai berikut:

1. Koneksi database dibuat menjadi satu file sendiri yaitu Koneksi.php, kemudian gunakan fungsi *require* ketika ingin melakukan operasi ke basis data.
2. Operasi data seperti Insert, Update, Delete, Get Data dibuat menjadi fungsi sendiri masing- masing dan disimpan di dalam satu file khusus yaitu Model.php

A. Source Code

MainActivity.kt

```
package com.example.listfilm

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.compose.rememberNavController
import com.example.listfilm.repository.MovieRepository
import com.example.listfilm.ui.screen.AppNavGraph
import com.example.listfilm.viewmodel.MovieViewModel
import com.example.listfilm.viewmodel.MovieViewModelFactory
import timber.log.Timber

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Timber.d("MainActivity onCreate() called")

        setContent {
            val navController = rememberNavController()
            val context = applicationContext
            val repository = MovieRepository(context)
            val viewModel: MovieViewModel = viewModel(
                factory = MovieViewModelFactory(repository)
            )
            AppNavGraph(navController = navController, viewModel =
viewModel)
        }
    }
}
```

Tabel 37.Source Code Soal 1 Modul 5

MovieDao.kt

```
package com.example.listfilm.data

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query

@Dao
interface MovieDao {
```

```

    @Query("SELECT * FROM movies")
    suspend fun getAllMovies(): List<MovieEntity>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertAll(movies: List<MovieEntity>)
}

```

Tabel 38.Source Code Soal 1 Modul 5

MovieDatabase.kt

```

package com.example.listfilm.data

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [MovieEntity::class], version = 1)
abstract class MovieDatabase : RoomDatabase() {
    abstract fun movieDao(): MovieDao

    companion object {
        @Volatile private var INSTANCE: MovieDatabase? = null

        fun getDatabase(context: Context): MovieDatabase {
            return INSTANCE ?: synchronized(this) {
                Room.databaseBuilder(
                    context.applicationContext,
                    MovieDatabase::class.java,
                    "movie_database"
                ).build().also { INSTANCE = it }
            }
        }
    }
}

```

Tabel 39.Source Code Soal 1 Modul 5

MovieEntity.kt

```

package com.example.listfilm.data

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "movies")
data class MovieEntity(
    @PrimaryKey val id: Int,
    val title: String,
    val posterPath: String,

```

```

        val overview: String
    )

```

Tabel 40.Source Code Soal 1 Modul 5

FilmListScreen.kt

```

package com.example.listfilm.ui.screen

import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.Card
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.collectAsState
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.navigation.NavHostController
import coil.compose.rememberAsyncImagePainter
import com.example.listfilm.model.Movie
import com.example.listfilm.util.Constants
import com.example.listfilm.viewmodel.MovieViewModel

@Composable
fun FilmListScreen(navController: NavHostController, viewModel: MovieViewModel)
{
    val context = LocalContext.current
    val movies = viewModel.movies.collectAsState().value

    LazyColumn(contentPadding = PaddingValues(16.dp)) {
        items(movies) { movie: Movie ->
            Card(
                shape = RoundedCornerShape(16.dp),
                modifier = Modifier
                    .padding(bottom = 16.dp)
                    .fillMaxWidth()
            ) {
                Column(Modifier.padding(16.dp)) {
                    Image(
                        painter
rememberAsyncImagePainter("${Constants.IMAGE_BASE_URL}${movie.posterPath}"),
                        contentDescription = null,
                        modifier = Modifier
                            .fillMaxWidth()
                            .height(200.dp)
                            .clip(RoundedCornerShape(12.dp)),
                        contentScale = ContentScale.Crop
                    )
                }
            }
        }
    }
}

```



```

        val movies = viewModel.movies.collectAsState().value

        NavHost(navController = navController, startDestination =
"list") {
            composable("list") {
                FilmListScreen(navController = navController,
viewModel = viewModel)
            }
            composable("detail/{id}") { backStackEntry ->
                val id =
backStackEntry.arguments?.getString("id")?.toIntOrNull()
                val movie = movies.find { it.id == id }
                if (movie != null) {
                    MovieDetailScreen(movie = movie, navController =
navController)
                } else {
                    Text("Movie not found", modifier =
Modifier.fillMaxSize())
                }
            }
        }
    }
}

```

MovieDetailScreen.kt

```

package com.example.listfilm.ui.screen

import androidx.compose.foundation.layout.*
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.navigation.NavHostController
import com.example.listfilm.model.Movie

@Composable
fun MovieDetailScreen(movie: Movie, navController:
NavHostController) {
    Column(modifier = Modifier
        .fillMaxSize()
        .padding(16.dp)) {

        Text(text = movie.title, style =
MaterialTheme.typography.headlineMedium)
        Spacer(modifier = Modifier.height(8.dp))
        Text(text = movie.overview)
    }
}

```

```

        Spacer(modifier = Modifier.height(16.dp))

        androidx.compose.material3.Button(onClick = {
            navController.popBackStack()
        }) {
            Text("Back")
        }
    }
}

```

Tabel 42.Source Code Soal 1 Modul 5

MovieViewModel.kt

```

package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.listfilm.model.Movie
import com.example.listfilm.repository.MovieRepository
import com.example.listfilm.util.Constants
import kotlinx.coroutines.flow.SharingStarted
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.stateIn
import timber.log.Timber

class MovieViewModel(private val repository: MovieRepository) :
    ViewModel() {
    val movies: StateFlow<List<Movie>> =
        repository.getPopularMovies(Constants.API_KEY)
            .stateIn(viewModelScope, SharingStarted.Eagerly,
emptyList())

    fun logDetailClicked(movie: Movie) {
        Timber.d("Detail clicked: ${movie.title}")
    }

    fun logOpenSiteClicked(movie: Movie) {
        Timber.d("Open site clicked: https://www.themoviedb.org/movie/${movie.id}")
    }
}

```

Tabel 43.Source Code Soal 1 Modul 5

ListFilmApp.kt

```

package com.example.listfilm

import android.app.Application
import timber.log.Timber

```

```

class ListFilmApp : Application() {
    override fun onCreate() {
        super.onCreate()
        Timber.plant(Timber.DebugTree())
    }
}

```

Tabel 44.Source Code Soal 1 Modul 5

MovieViewModelFactory.kt

```

package com.example.listfilm.viewmodel

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import com.example.listfilm.repository.MovieRepository

class MovieViewModelFactory(
    private val repository: MovieRepository
) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T
    {
        if
        (modelClass.isAssignableFrom(MovieViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
            return MovieViewModel(repository) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}

```

Tabel 45.Source Code Soal 1 Modul 5

Constants.kt

```

package com.example.listfilm.util

object Constants {
    const val BASE_URL = "https://api.themoviedb.org/3/"
    const val IMAGE_BASE_URL = "https://image.tmdb.org/t/p/w500"
    const val API_KEY = "8daab9d30b9168bb81cb48d5fcd1f302"
}

```

Tabel 46.Source Code Soal 1 Modul 5

MovieRespository.kt

```

package com.example.listfilm.repository

import android.content.Context
import com.example.listfilm.data.MovieDatabase

```



```

import com.example.listfilm.data.MovieEntity
import com.example.listfilm.model.Movie
import com.example.listfilm.network.RetrofitInstance
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.flow
import timber.log.Timber

class MovieRepository(context: Context) {
    private val movieDao =
        MovieDatabase.getDatabase(context).movieDao()

    fun getPopularMovies(apiKey: String): Flow<List<Movie>> =
        flow {
            try {
                val response =
                    RetrofitInstance.api.getPopularMovies(apiKey)
                Timber.d("Ambil dari API berhasil:
                    ${response.results.size}")
                val movieEntities = response.results.map {
                    MovieEntity(it.id, it.title, it.posterPath,
                    it.overview)
                }
                movieDao.insertAll(movieEntities)
                emit(response.results) // from API
            } catch (e: Exception) {
                Timber.e("Gagal ambil dari API: ${e.message}")
                val cached = movieDao.getAllMovies().map {
                    Movie(it.id, it.title, it.posterPath,
                    it.overview)
                }
                emit(cached) // fallback to local Room
            }
        }
}

```

Tabel 47.Source Code Soal 1 Modul 5

PreferenceManager.kt

```

package com.example.listfilm.data.preferences

import android.content.Context
import android.content.SharedPreferences

class PreferenceManager(context: Context) {
    private val prefs: SharedPreferences =
        context.getSharedPreferences("user_prefs",
        Context.MODE_PRIVATE)
}

```

```

    fun setDarkMode(enabled: Boolean) {
        prefs.edit().putBoolean("dark_mode", enabled).apply()
    }

    fun isDarkMode(): Boolean = prefs.getBoolean("dark_mode",
false)
}

```

Tabel 48.Source Code Soal 1 Modul 5

Movie.kt

```

package com.example.listfilm.model

import kotlinx.serialization.SerialName
import kotlinx.serialization.Serializable

@Serializable
data class Movie(
    val id: Int,
    val title: String,
    @SerialName("poster_path") val posterPath: String,
    val overview: String
)

```

Tabel 49.Source Code Soal 1 Modul 5

MovieResponse.kt

```

package com.example.listfilm.model

import kotlinx.serialization.Serializable

@Serializable
data class MovieResponse(
    val page: Int,
    val results: List<Movie>
)

```

Tabel 50.Source Code Soal 1 Modul 5

RetrofitInstance.kt

```

package com.example.listfilm.network

import
com.jakewharton.retrofit2.converter.kotlinx.serialization.asConverterFactory
import kotlinx.serialization.json.Json
import okhttp3.MediaType.Companion.toMediaType
import retrofit2.Retrofit

```

```

object RetrofitInstance {
    private val json = Json { ignoreUnknownKeys = true }
    private val contentType = "application/json".toMediaType()

    val api: TmdbApiService by lazy {
        Retrofit.Builder()
            .baseUrl("https://api.themoviedb.org/3/")
            .addConverterFactory(json.asConverterFactory(contentType))
            .build()
            .create(TmdbApiService::class.java)
    }
}

```

Tabel 51.Source Code Soal 1 Modul 5

TmdbApiService.kt

```

package com.example.listfilm.network

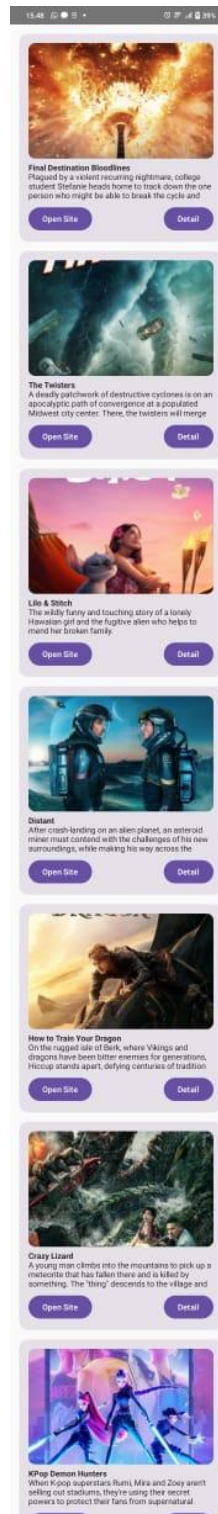
import com.example.listfilm.model.MovieResponse
import retrofit2.http.GET
import retrofit2.http.Query

interface TmdbApiService {
    @GET("movie/popular")
    suspend fun getPopularMovies(
        @Query("api_key") apiKey: String
    ): MovieResponse
}

```

Tabel 52.Source Code Soal 1 Modul 5

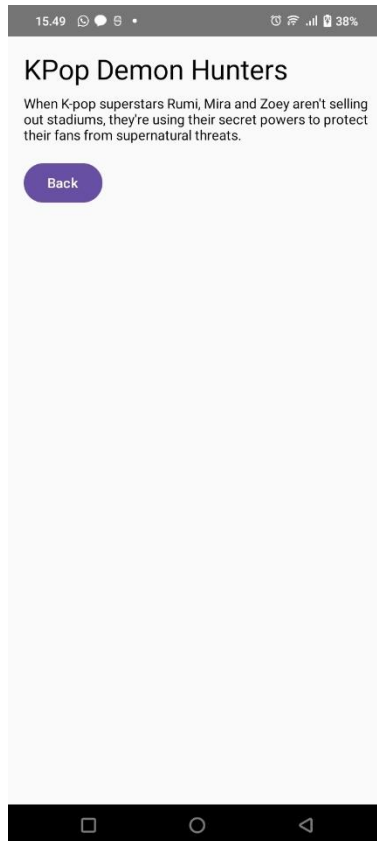
B. Output Program



Gambar 33.Screenshot Hasil Jawaban Soal 1 Modul 5



Gambar 34. Screenshot Hasil Jawaban Soal 1 Modul 5



Gambar 35. Screenshot Hasil Jawaban Soal 1 Modul

C. Pembahasan

Aplikasi ListFilm merupakan aplikasi Android berbasis Jetpack Compose yang dibangun dengan arsitektur MVVM (Model-View-ViewModel). Titik awal aplikasi berada di MainActivity, di mana UI Compose diinisialisasi menggunakan setContent. Di dalamnya, NavController dibuat untuk navigasi antar layar, MovieRepository diinstansiasi, dan MovieViewModel disiapkan melalui MovieViewModelFactory. ViewModel ini bertugas mengelola data dan logika bisnis aplikasi, termasuk mengambil data dari repository serta mencatat aktivitas pengguna melalui Timber. Repository sendiri menjadi jembatan antara data yang berasal dari API (TheMovieDB) dan database lokal yang dikelola dengan Room. Saat data berhasil diambil dari API menggunakan Retrofit, data tersebut disimpan ke database lokal. Jika pengambilan data dari API gagal, repository akan menampilkan data dari database lokal sebagai fallback.

Untuk kebutuhan pengambilan data, digunakan Retrofit yang dikonfigurasi melalui RetrofitInstance, dengan dukungan KotlinX Serialization untuk parsing JSON. Endpoint API didefinisikan dalam TmdbApiService. Model data dari API disimpan dalam kelas Movie dan MovieResponse, yang masing-masing dilengkapi anotasi @Serializable. Sementara itu, untuk penyimpanan lokal, digunakan entitas Room MovieEntity yang diakses melalui MovieDao, dan dikelola oleh MovieDatabase yang menerapkan pola singleton. Data ini digunakan oleh ViewModel dan disalurkan ke UI melalui StateFlow.

Navigasi antar layar ditangani oleh AppNavGraph, yang mengatur dua layar utama: FilmListScreen dan MovieDetailScreen. FilmListScreen menampilkan daftar film menggunakan LazyColumn, dan setiap film ditampilkan dalam kartu yang menyertakan tombol untuk membuka detail dan mengunjungi situs TMDB. Jika pengguna memilih film tertentu, aplikasi akan berpindah ke MovieDetailScreen, yang menampilkan informasi lengkap tentang film tersebut dan menyediakan tombol kembali ke daftar. Untuk pengelolaan preferensi pengguna, seperti mode gelap, aplikasi menyediakan kelas PreferenceManager yang memanfaatkan SharedPreferences.

Seluruh aplikasi mengandalkan ListFilmApp sebagai kelas Application untuk inisialisasi global, khususnya untuk mengaktifkan Timber sebagai logger. Dengan komponen-komponen ini, aplikasi ListFilm memiliki alur data yang terstruktur, dukungan caching lokal, navigasi yang jelas, serta desain UI modern yang didukung oleh Jetpack Compose. Aplikasi ini cukup lengkap untuk digunakan sebagai dasar pengembangan proyek menengah hingga lanjut.

Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.

<https://github.com/rc114/PrakMobile.git>